Algorithms

Lecture 7

Algorithms

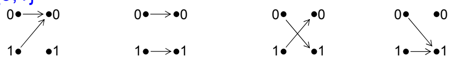
- Deutsch's algorithm: $\{0,1\} \rightarrow \{0,1\}$
- Deutsch-Jozsa algorithm: $\{0,1\}^n \rightarrow \{0,1\}$
- Simon's periodicity algorithm: $\{0,1\}^n \rightarrow \{0,1\}^n$
- Grover's search algorithm: unordered array of size n in $\sqrt{n}$ time instead of n time
- Shor's factoring algorithm: factor numbers in polynomial time.

Basic steps in a quantum algorithm

- All quantum algorithms:
  - The system will start with the qubits in a particular classical state.
  - The system is put into a superposition of many states.
  - Acting on this superposition with several unitary operations.
  - A measurement of the qubits

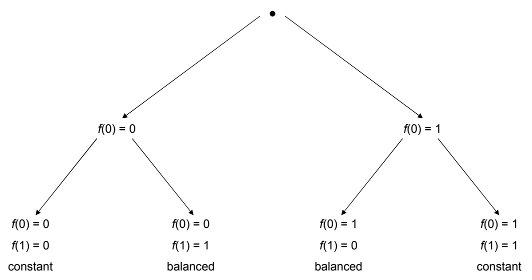
Deutsch's algorithm

- Simplest quantum algorithm
- Concerned with functions from the set {0,1} to the set {0,1}



- A function f: $\{0,1\} \rightarrow \{0,1\}$ is balanced if $f(0) \neq f(1)$, i.e. it is one to one; in contrast it is constant if $f(0) = f(1)$.
- Deutsch's algorithm: given a function f: $\{0,1\} \rightarrow \{0,1\}$ as a black box, where one can evaluate an input, but cannot "look inside" and "see" how the function is defined, determine if the function is balanced or constant.

Classical computer

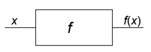


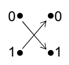
With a classical computer f must be evaluate twice; can we do better on a quantum computer?

A quantum computer can be in a superposition of two basic states at the same time.

Evaluation of a function

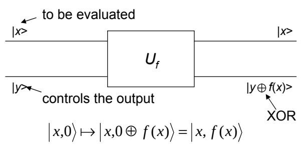
- Classical:





$$\leftrightarrow \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 0 \end{array}$$

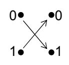
- Quantum system
  - Unitary (reversible)



$$|x,0\rangle \mapsto |x,0 \oplus f(x)\rangle = |x, f(x)\rangle$$

U_f is its own reverse:

$$|x,y\rangle \mapsto |x, y \oplus f(x)\rangle$$
$$\mapsto |x,(y \oplus f(x)) \oplus f(x)\rangle = |x, y \oplus (f(x) \oplus f(x))\rangle = |x, y \oplus 0\rangle = |x,y\rangle$$



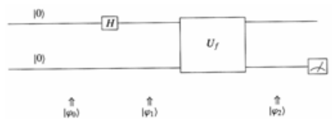
$$\leftrightarrow \quad U_f = \begin{array}{c|cccc} & 00 & 01 & 10 & 11 \\ \hline 00 & 0 & 1 & 0 & 0 \\ 01 & 1 & 0 & 0 & 0 \\ 10 & 0 & 0 & 1 & 0 \\ 11 & 0 & 0 & 0 & 1 \end{array}$$

Quantum “trick”

- Rather than evaluating *f* twice, put the top input in superposition: $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$
- This can be achieved by the Hadamard matrix:

$$H|0\rangle=\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}}\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}=\begin{bmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{bmatrix}=\frac{|0\rangle+|1\rangle}{\sqrt{2}}$$

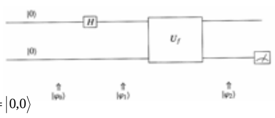
- Following quantum circuit:



- In terms of matrices:

$$U_f(H\otimes I)(|0\rangle\otimes|0\rangle)=U_f(H\otimes I)(|0,0\rangle)$$

Quantum “trick” (cont’d)



The system starts in $|\varphi_0\rangle=|0\rangle\otimes|0\rangle=|0,0\rangle$

Apply Hadamard matrix on top input $|\varphi_1\rangle=\left[\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right]|0\rangle=\frac{|0,0\rangle+|1,0\rangle}{\sqrt{2}}$

Multiplying with *U*U_f $|\varphi_2\rangle=\frac{|0,f(0)\rangle+|1,f(1)\rangle}{\sqrt{2}}$

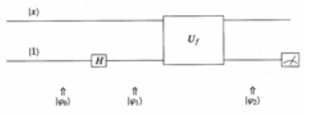
If we measure the top qubit, there will be a 50-50% chance of finding it in state |0> and a 50-50% chance of finding it in state |1>.

Similarly, there is no real information to be gotten by measuring the bottom qubit.

So the obvious algorithm does not work, we need a better trick!

Better “trick”

- Put the bottom qubit in the superposition state $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$, notice the minus sign!
- Quantum circuit:



- In terms of matrices: $U_f(I\otimes H)(|x,1\rangle)$
- Start with $|\varphi_0\rangle=|x,1\rangle$
- After the Hadamard matrix $|\varphi_1\rangle=|x\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]=\frac{|x,0\rangle-|x,1\rangle}{\sqrt{2}}$
- Applying U_f

$$|\varphi_2\rangle=|x\rangle\left[\frac{|0\oplus f(x)\rangle-|1\oplus f(x)\rangle}{\sqrt{2}}\right]=|x\rangle\left[\frac{|f(x)\rangle-|\overline{f(x)}\rangle}{\sqrt{2}}\right]=\begin{cases}|x\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right], \text{if } f(x)=0\\ |x\rangle\left[\frac{|1\rangle-|0\rangle}{\sqrt{2}}\right], \text{if } f(x)=1\end{cases}$$

Better “trick” (cont’d)

$$|\varphi_2\rangle=\begin{cases}|x\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right], \text{if } f(x)=0\\ |x\rangle\left[\frac{|1\rangle-|0\rangle}{\sqrt{2}}\right], \text{if } f(x)=1\end{cases}$$

with (*a*–*b*)=(–1)(*b*–*a*)

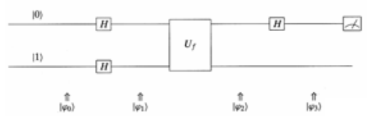
$$|\varphi_2\rangle=(-1)^{f(x)}|x\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$$

Evaluate top or bottom state?

No information: top qubit will be in state |*x*> and the bottom qubit either in state |0> or in state |1>……

Deutsch’s algorithm

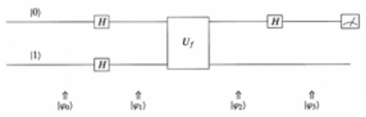
- Combine both “tricks”:
  - Both top and bottom qubits in superposition
  - Result of top qubit through Hadamard matrix



- In terms of matrices:

$$(H\otimes I)U_f(H\otimes H)|0,1\rangle \quad \text{or} \quad (H\otimes I)U_f(H\otimes H)\begin{array}{c}00\\01\\10\\11\end{array}\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$$

Deutsch’s algorithm (cont’d)



- Start with $|\varphi_0\rangle=|0,1\rangle$
- and $|\varphi_1\rangle=\left[\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]=\frac{+|0,0\rangle-|0,1\rangle+|1,0\rangle-|1,1\rangle}{2}=\begin{array}{c}00\\01\\10\\11\end{array}\begin{bmatrix}+\frac{1}{2}\\-\frac{1}{2}\\+\frac{1}{2}\\-\frac{1}{2}\end{bmatrix}$
- We saw that with bottom qubit in superposition and then multiply by U_f $(-1)^{f(x)}|x\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$
- with |*x*> in a superposition, we have

$$|\varphi_2\rangle=\left[\frac{(-1)^{f(0)}|0\rangle+(-1)^{f(1)}|1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$$

Deutsch's algorithm (cont'd)

- We have $|\varphi_2\rangle = \left[\frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$
- Let have a look at $(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle$
  - if *f* is constant $+1(|0\rangle + |1\rangle)$ or $-1(|0\rangle + |1\rangle)$ (constantly 0 or 1, resp.)
  - if *f* is balanced $+1(|0\rangle - |1\rangle)$ or $-1(|0\rangle - |1\rangle)$
- So we have $|\varphi_2\rangle = \begin{cases} (\pm 1)\left[\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right], \text{if } f \text{ is constant} \\ (\pm 1)\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]\left[\frac{|1\rangle-|0\rangle}{\sqrt{2}}\right], \text{if } f \text{ is balanced} \end{cases}$
- Hadamard matrix is its own reverse $\frac{|0\rangle+|1\rangle}{\sqrt{2}} \mapsto |0\rangle$ and $\frac{|0\rangle-|1\rangle}{\sqrt{2}} \mapsto |1\rangle$
- Apply it to top qubit

$$|\varphi_3\rangle = \begin{cases} (\pm 1)|0\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right], \text{if } f \text{ is constant} \\ (\pm 1)|1\rangle\left[\frac{|1\rangle-|0\rangle}{\sqrt{2}}\right], \text{if } f \text{ is balanced} \end{cases}$$

- Measure top qubit: if |0> then *f* is constant, otherwise balanced.
  Only one evaluation of *f*.

Deutsch's algorithm (cont'd)

Remarks:

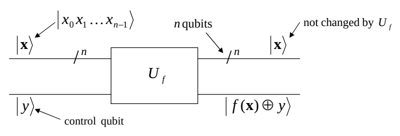
- The ±1 tells us which of the two balanced or constant functions we have, but can not be measured.
- Output of top qubit of U_f not the same as the input: inclusion of Hadamard matrices makes top and bottom qubits entangled.
- Trick? No changing around the information:
  1. Is the function balanced or constant?
  2. What is the value of the function on 0?

Deutsch-Jozsa algorithm

- Generalization:
  - *f* : $\{0,1\}^n \rightarrow \{0,1\}$, which accepts a string of *n* 0's and 1's (natural numbers from 0 to 2^{n-1}) and outputs a zero or one.
  - *f* is called balanced if exactly half of the inputs go to 0 (and the other half go to 1).
  - *f* is called constant if *all* the inputs go to 0 or *all* the inputs go to 1.
- Problem:
  - Given a function of $\{0,1\}^n$ to $\{0,1\}$, which you can evaluate but cannot "see" the way it is defined.
  - The function is either balanced or constant.
  - Determine if the function is balanced or constant.
  - *n*=1: Deutsch algorithm.
- Classically:
  - Evaluate the function on different inputs.
  - Best scenario: first two different inputs have different outputs → balanced function.
  - Worst scenario: $2^n/2+1 = 2^{n-1}+1$ evaluations.

Solution: superposition

- In Deutsch's algorithm we used the superposition of two possible input states. Now we enter a superposition of all 2^n possible input states.



Tensor product of Hadamard matrices

- Single qubit in superposition: single Hadamard matrix; *n* qubits in superposition: tensor product of *n* Hadamard matrices:

$$H, H \otimes H = H^{\otimes 2}, H \otimes H \otimes H = H^{\otimes 3}, \dots, H^{\otimes n}$$

- Hadamard matrix definition:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \text{ or } H[i,j] = \tfrac{1}{\sqrt{2}}(-1)^{i \wedge j}: \; H = \frac{1}{\sqrt{2}}\begin{bmatrix} (-1)^{0\wedge 0} & (-1)^{0\wedge 1} \\ (-1)^{1\wedge 0} & (-1)^{1\wedge 1} \end{bmatrix}$$

0 and 1 as Boolean values, and $(-1)^0=1$ and $(-1)^1=-1$.

Tensor product of Hadamard matrices (cont'd)

- We can calculate

$$H^{\otimes 2} = H \otimes H = \frac{1}{\sqrt{2}}\begin{bmatrix} (-1)^{0\wedge 0} & (-1)^{0\wedge 1} \\ (-1)^{1\wedge 0} & (-1)^{1\wedge 1} \end{bmatrix} \otimes \frac{1}{\sqrt{2}}\begin{bmatrix} (-1)^{0\wedge 0} & (-1)^{0\wedge 1} \\ (-1)^{1\wedge 0} & (-1)^{1\wedge 1} \end{bmatrix}$$

$$= \frac{1}{\sqrt{2}} * \frac{1}{\sqrt{2}}\begin{bmatrix} (-1)^{0\wedge 0} * (-1)^{0\wedge 0} & (-1)^{0\wedge 0} * (-1)^{0\wedge 1} & (-1)^{0\wedge 1} * (-1)^{0\wedge 0} & (-1)^{0\wedge 1} * (-1)^{0\wedge 1} \\ (-1)^{0\wedge 0} * (-1)^{1\wedge 0} & (-1)^{0\wedge 0} * (-1)^{1\wedge 1} & (-1)^{0\wedge 1} * (-1)^{1\wedge 0} & (-1)^{0\wedge 1} * (-1)^{1\wedge 1} \\ (-1)^{1\wedge 0} * (-1)^{0\wedge 0} & (-1)^{1\wedge 0} * (-1)^{0\wedge 1} & (-1)^{1\wedge 1} * (-1)^{0\wedge 0} & (-1)^{1\wedge 1} * (-1)^{0\wedge 1} \\ (-1)^{1\wedge 0} * (-1)^{1\wedge 0} & (-1)^{1\wedge 0} * (-1)^{1\wedge 1} & (-1)^{1\wedge 1} * (-1)^{1\wedge 0} & (-1)^{1\wedge 1} * (-1)^{1\wedge 1} \end{bmatrix}$$

- We are not interested in $(-1)^{x+y}$, but in the parity of *x* and *y* (exclusive-or):

$$H^{\otimes 2} = \frac{1}{2}\begin{bmatrix} (-1)^{0\wedge 0 \oplus 0\wedge 0} & (-1)^{0\wedge 0 \oplus 0\wedge 1} & (-1)^{0\wedge 1 \oplus 0\wedge 0} & (-1)^{0\wedge 1 \oplus 0\wedge 1} \\ (-1)^{0\wedge 0 \oplus 1\wedge 0} & (-1)^{0\wedge 0 \oplus 1\wedge 1} & (-1)^{0\wedge 1 \oplus 1\wedge 0} & (-1)^{0\wedge 1 \oplus 1\wedge 1} \\ (-1)^{1\wedge 0 \oplus 0\wedge 0} & (-1)^{1\wedge 0 \oplus 0\wedge 1} & (-1)^{1\wedge 1 \oplus 0\wedge 0} & (-1)^{1\wedge 1 \oplus 0\wedge 1} \\ (-1)^{1\wedge 0 \oplus 1\wedge 0} & (-1)^{1\wedge 0 \oplus 1\wedge 1} & (-1)^{1\wedge 1 \oplus 1\wedge 0} & (-1)^{1\wedge 1 \oplus 1\wedge 1} \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Tensor product of Hadamard matrices (cont'd)

- Proved by induction that the scalar coefficient of $H^{\otimes n}$ is $\frac{1}{\sqrt{2^n}} = 2^{-\frac{n}{2}}$
- Useful operation $\langle\ ,\ \rangle : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$

  Definition : given two binary strings of length n, $\mathbf{x} = x_0x_1x_2\ldots x_{n-1}$ and $\mathbf{y} = y_0y_1y_2\ldots y_{n-1}$, we have

  $$\langle \mathbf{x},\mathbf{y}\rangle = \langle x_0x_1x_2\ldots x_{n-1}, y_0y_1y_2\ldots y_{n-1}\rangle$$
  $$= (x_0 \wedge y_0) \oplus (x_1 \wedge y_1) \oplus \cdots \oplus (x_{n-1} \wedge y_{n-1})$$

  – Basically it gives the parity of the number of times that both bits are 1.
- If **x** and **y** are binary strings of length *n*, then **x**⊕**y** is the pointwise (bitwise) exclusive-or operation

  $$\mathbf{x} \oplus \mathbf{y} = x_0 \oplus y_0, x_1 \oplus y_1, \ldots, x_{n-1} \oplus y_{n-1}$$

$H^{\otimes 3}$

$$\frac{1}{2\sqrt{2}}\begin{array}{c|cccccccc}
 & \mathbf{000} & \mathbf{001} & \mathbf{010} & \mathbf{011} & \mathbf{100} & \mathbf{101} & \mathbf{110} & \mathbf{111} \\
\mathbf{000} & (-1)^{\langle 000,000\rangle} & (-1)^{\langle 000,001\rangle} & (-1)^{\langle 000,010\rangle} & (-1)^{\langle 000,011\rangle} & (-1)^{\langle 000,100\rangle} & (-1)^{\langle 000,101\rangle} & (-1)^{\langle 000,110\rangle} & (-1)^{\langle 000,111\rangle} \\
\mathbf{001} & (-1)^{\langle 001,000\rangle} & (-1)^{\langle 001,001\rangle} & (-1)^{\langle 001,010\rangle} & (-1)^{\langle 001,011\rangle} & (-1)^{\langle 001,100\rangle} & (-1)^{\langle 001,101\rangle} & (-1)^{\langle 001,110\rangle} & (-1)^{\langle 001,111\rangle} \\
\mathbf{010} & (-1)^{\langle 010,000\rangle} & (-1)^{\langle 010,001\rangle} & (-1)^{\langle 010,010\rangle} & (-1)^{\langle 010,011\rangle} & (-1)^{\langle 010,100\rangle} & (-1)^{\langle 010,101\rangle} & (-1)^{\langle 010,110\rangle} & (-1)^{\langle 010,111\rangle} \\
\mathbf{011} & (-1)^{\langle 011,000\rangle} & (-1)^{\langle 011,001\rangle} & (-1)^{\langle 011,010\rangle} & (-1)^{\langle 011,011\rangle} & (-1)^{\langle 011,100\rangle} & (-1)^{\langle 011,101\rangle} & (-1)^{\langle 011,110\rangle} & (-1)^{\langle 011,111\rangle} \\
\mathbf{100} & (-1)^{\langle 100,000\rangle} & (-1)^{\langle 100,001\rangle} & (-1)^{\langle 100,010\rangle} & (-1)^{\langle 100,011\rangle} & (-1)^{\langle 100,100\rangle} & (-1)^{\langle 100,101\rangle} & (-1)^{\langle 100,110\rangle} & (-1)^{\langle 100,111\rangle} \\
\mathbf{101} & (-1)^{\langle 101,000\rangle} & (-1)^{\langle 101,001\rangle} & (-1)^{\langle 101,010\rangle} & (-1)^{\langle 101,011\rangle} & (-1)^{\langle 101,100\rangle} & (-1)^{\langle 101,101\rangle} & (-1)^{\langle 101,110\rangle} & (-1)^{\langle 101,111\rangle} \\
\mathbf{110} & (-1)^{\langle 110,000\rangle} & (-1)^{\langle 110,001\rangle} & (-1)^{\langle 110,010\rangle} & (-1)^{\langle 110,011\rangle} & (-1)^{\langle 110,100\rangle} & (-1)^{\langle 110,101\rangle} & (-1)^{\langle 110,110\rangle} & (-1)^{\langle 110,111\rangle} \\
\mathbf{111} & (-1)^{\langle 111,000\rangle} & (-1)^{\langle 111,001\rangle} & (-1)^{\langle 111,010\rangle} & (-1)^{\langle 111,011\rangle} & (-1)^{\langle 111,100\rangle} & (-1)^{\langle 111,101\rangle} & (-1)^{\langle 111,110\rangle} & (-1)^{\langle 111,111\rangle}
\end{array}$$

$$= \frac{1}{2\sqrt{2}}\begin{array}{c|cccccccc}
 & \mathbf{000} & \mathbf{001} & \mathbf{010} & \mathbf{011} & \mathbf{100} & \mathbf{101} & \mathbf{110} & \mathbf{111} \\
\mathbf{000} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\mathbf{001} & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
\mathbf{010} & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
\mathbf{011} & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
\mathbf{100} & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
\mathbf{101} & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
\mathbf{110} & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
\mathbf{111} & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1
\end{array}.$$

Tensor product of Hadamard matrices (cont'd)

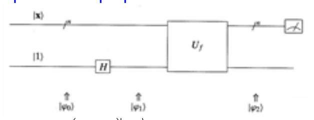
- General formula

  $H^{\otimes n}[\mathbf{i},\mathbf{j}] = \frac{1}{\sqrt{2^n}}(-1)^{\langle \mathbf{i},\mathbf{j}\rangle}$, where **i** and **j** are the row and column numbers in binary.
- What happens if we multiply a state with this matrix? Notice all elements of the leftmost column of $H^{\otimes n}$ are +1. So if we multiply with the state |**0**> = |00…0> = [1,0,…,0]^T this will be equal to the leftmost column of $H^{\otimes n}$:

$$H^{\otimes n}|\mathbf{0}\rangle = H^{\otimes n}[-,\mathbf{0}] = \frac{1}{\sqrt{2^n}}\begin{array}{c}00000000\\00000001\\00000010\\\vdots\\11111110\\11111111\end{array}\begin{bmatrix}1\\1\\1\\\vdots\\1\\1\end{bmatrix} = \frac{1}{\sqrt{2^n}}\sum_{\mathbf{x}\in\{0,1\}^n}|\mathbf{x}\rangle$$

Deutsch-Jozsa algorithm

- Bottom control qubit in a superposition:

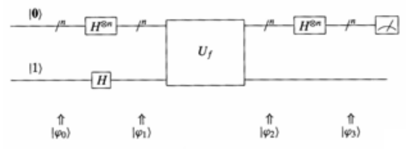


- In terms of matrices $U_f(I \otimes H)|\mathbf{x},1\rangle$
- We start with $|\varphi_0\rangle = |\mathbf{x},1\rangle$
- After the bottom Hadamard matrix $|\varphi_1\rangle = |\mathbf{x}\rangle\left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] = \frac{|\mathbf{x},0\rangle - |\mathbf{x},1\rangle}{\sqrt{2}}$
- Applying U_f $|\varphi_2\rangle = |\mathbf{x}\rangle\left[\frac{|f(\mathbf{x})\oplus 0\rangle - |f(\mathbf{x})\oplus 1\rangle}{\sqrt{2}}\right] = |\mathbf{x}\rangle\left[\frac{|f(\mathbf{x})\rangle - |\overline{f(\mathbf{x})}\rangle}{\sqrt{2}}\right]$

  $$= \begin{cases}|\mathbf{x}\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right], \text{if } f(\mathbf{x}) = 0\\ |\mathbf{x}\rangle\left[\frac{|1\rangle-|0\rangle}{\sqrt{2}}\right], \text{if } f(\mathbf{x}) = 1\end{cases} = (-1)^{f(\mathbf{x})}|\mathbf{x}\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$$
- Useless!

Deutsch-Jozsa algorithm (cont'd)

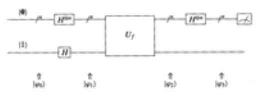
- Put |**x**> into a superposition in which all 2^n possible strings have equal probability



- In terms of matrices $(H^{\otimes n} \otimes I)U_f(H^{\otimes n} \otimes H)|\mathbf{0},1\rangle$

Deutsch-Jozsa algorithm (cont'd)

$(H^{\otimes n} \otimes I)U_f(H^{\otimes n} \otimes H)|\mathbf{0},1\rangle$



We start with $|\varphi_0\rangle = |\mathbf{0},1\rangle$

Then $|\varphi_1\rangle = \left[\frac{\sum_{\mathbf{x}\in\{0,1\}^n}|\mathbf{x}\rangle}{\sqrt{2^n}}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$

Applying U_f $|\varphi_2\rangle = \left[\frac{\sum_{\mathbf{x}\in\{0,1\}^n}(-1)^{f(\mathbf{x})}|\mathbf{x}\rangle}{\sqrt{2^n}}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$

Make a superposition of a superposition on the top qubits

$$|\varphi_3\rangle = \left[\frac{\sum_{\mathbf{x}\in\{0,1\}^n}(-1)^{f(\mathbf{x})}\sum_{\mathbf{z}\in\{0,1\}^n}(-1)^{\langle \mathbf{z},\mathbf{x}\rangle}|\mathbf{z}\rangle}{2^n}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$$

Deutsch-Jozsa algorithm (cont'd)

$$|\varphi_3\rangle = \left[\frac{\sum_{\mathbf{x}\in\{0,1\}^n}(-1)^{f(\mathbf{x})}\sum_{\mathbf{z}\in\{0,1\}^n}(-1)^{\langle\mathbf{z},\mathbf{x}\rangle}|\mathbf{z}\rangle}{2^n}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$$
$$= \left[\frac{\sum_{\mathbf{x}\in\{0,1\}^n}\sum_{\mathbf{z}\in\{0,1\}^n}(-1)^{f(\mathbf{x})}(-1)^{\langle\mathbf{z},\mathbf{x}\rangle}|\mathbf{z}\rangle}{2^n}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$$
$$= \left[\frac{\sum_{\mathbf{x}\in\{0,1\}^n}\sum_{\mathbf{z}\in\{0,1\}^n}(-1)^{f(\mathbf{x})\otimes\langle\mathbf{z},\mathbf{x}\rangle}|\mathbf{z}\rangle}{2^n}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$$

Measure top qubit of φ_3; what is the probability that it will collapse to state |**0**>?

Answer: set **z** = **0** and realize that <**z**,**x**> = <**0**,**x**> = 0 for all x. Then

$$|\varphi_3\rangle = \left[\frac{\sum_{\mathbf{x}\in\{0,1\}^n}(-1)^{f(\mathbf{x})}|\mathbf{0}\rangle}{2^n}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$$

Deutsch-Jozsa algorithm (cont'd)

$$|\varphi_3\rangle = \left[\frac{\sum_{\mathbf{x}\in\{0,1\}^n}(-1)^{f(\mathbf{x})}|\mathbf{0}\rangle}{2^n}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$$

Probability of collapsing to |**0**> is totally dependent on *f* (**x**).

If *f* (**x**) is constant 1, the top qubits become

$$\frac{\sum_{\mathbf{x}\in\{0,1\}^n}(-1)|\mathbf{0}\rangle}{2^n} = \frac{-(2^n)|\mathbf{0}\rangle}{2^n} = -1|\mathbf{0}\rangle$$

If *f* (**x**) is constant 0, the top qubits become

$$\frac{\sum_{\mathbf{x}\in\{0,1\}^n}1|\mathbf{0}\rangle}{2^n} = \frac{2^n|\mathbf{0}\rangle}{2^n} = +1|\mathbf{0}\rangle$$

If *f* (**x**) is balanced, then half of the **x**'s will cancel the other half and the top qubits become

$$\frac{\sum_{\mathbf{x}\in\{0,1\}^n}(-1)^{f(\mathbf{x})}|\mathbf{0}\rangle}{2^n} = \frac{0|\mathbf{0}\rangle}{2^n} = 0|\mathbf{0}\rangle$$

We only get |**0**> if the function is constant. If anything else is measured, then the function is balanced.

Only one function evaluation instead of 2^{n-1}: exponential speedup!

Simon's periodicity algorithm

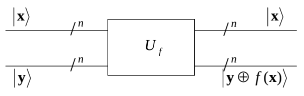
- Finding patterns in functions.
- Given a function $f : \{0,1\}^n \to \{0,1\}^n$ that we can evaluate, but is given as a black box.
- There is a secret (hidden) binary string **c** = $c_0c_1c_2\ldots c_{n-1}$, such that for all strings **x**, **y** we have
$$f(\mathbf{x}) = f(\mathbf{y}) \quad \text{if and only if} \quad \mathbf{x} = \mathbf{y} \oplus \mathbf{c}$$
- In other words, the values of *f* repeat themselves in some pattern, and the pattern is determined by **c**, the period of *f*.
- Goal of Simon's algorithm is to determine **c**.

Example

- Let *n* = 3. Consider **c** = 101. Then we have the following requirements on *f*:

$$000 \oplus 101 = 101; \text{hence, } f(000) = f(101).$$
$$001 \oplus 101 = 100; \text{hence, } f(001) = f(100).$$
$$010 \oplus 101 = 111; \text{hence, } f(010) = f(111).$$
$$011 \oplus 101 = 110; \text{hence, } f(011) = f(110).$$
$$100 \oplus 101 = 001; \text{hence, } f(100) = f(001).$$
$$101 \oplus 101 = 000; \text{hence, } f(101) = f(000).$$
$$110 \oplus 101 = 011; \text{hence, } f(110) = f(011).$$
$$111 \oplus 101 = 010; \text{hence, } f(111) = f(010).$$

- Notice that if **c** = 0^n, then the function is one to one; otherwise it is two to one.

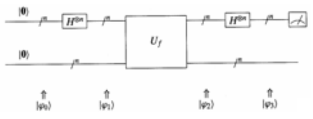


Classically

- Evaluate *f* on different binary strings.
- After each evaluation, check if the output has already been found.
- If for two input $\mathbf{x}_1$ and $\mathbf{x}_2$ holds $f(\mathbf{x}_1) = f(\mathbf{x}_2)$ then
$$\mathbf{x}_1 = \mathbf{x}_2 \oplus \mathbf{c}$$
- and can **c** be obtained by
$$\mathbf{x}_1 \oplus \mathbf{x}_2 = \mathbf{x}_2 \oplus \mathbf{c} \oplus \mathbf{x}_2 = \mathbf{c}$$
- If the function is two-to-one, we do not have to evaluate more than half the inputs before we get a repeat. If we have to evaluate more, we know **c** = 0^n. So, the worst case is $2^n/2 + 1 = 2^{n-1} + 1$.
- Can we do better?

Quantum version

- Performing the following operations several times:



- We start with $|\varphi_0\rangle = |\mathbf{0},\mathbf{0}\rangle$
- Put the input in a superposition of all possible inputs
$$|\varphi_1\rangle = \left[\frac{\sum_{\mathbf{x}\in\{0,1\}^n}|\mathbf{x},\mathbf{0}\rangle}{\sqrt{2^n}}\right]$$
- Evaluation of *f* on all these possibilities
$$|\varphi_2\rangle = \left[\frac{\sum_{\mathbf{x}\in\{0,1\}^n}|\mathbf{x}, f(\mathbf{x})\rangle}{\sqrt{2^n}}\right]$$
- Apply *n* Hadamard tensor product
$$|\varphi_3\rangle = \frac{\sum_{\mathbf{x}\in\{0,1\}^n}\sum_{\mathbf{z}\in\{0,1\}^n}(-1)^{\langle\mathbf{z},\mathbf{x}\rangle}|\mathbf{z}, f(\mathbf{x})\rangle}{2^n}$$

Quantum version (cont’d)

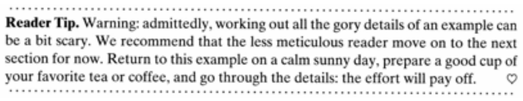
- For each input **x** and for each **z**, we know that the following kets are equal

  $|\mathbf{z}, f(\mathbf{x})\rangle$ and $|\mathbf{z}, f(\mathbf{x} \oplus \mathbf{c})\rangle$
- The coefficient for this ket is $\frac{(-1)^{\langle \mathbf{z},\mathbf{x}\rangle} + (-1)^{\langle \mathbf{z},\mathbf{x}\oplus\mathbf{c}\rangle}}{2^n}$
- <–,–> is an inner product, so $\frac{(-1)^{\langle \mathbf{z},\mathbf{x}\rangle} + (-1)^{\langle \mathbf{z},\mathbf{x}\oplus\mathbf{c}\rangle}}{2^n} = \frac{(-1)^{\langle \mathbf{z},\mathbf{x}\rangle} + (-1)^{\langle \mathbf{z},\mathbf{x}\rangle\oplus\langle \mathbf{z},\mathbf{c}\rangle}}{2^n}$

  $= \frac{(-1)^{\langle \mathbf{z},\mathbf{x}\rangle} + (-1)^{\langle \mathbf{z},\mathbf{x}\rangle}(-1)^{\langle \mathbf{z},\mathbf{c}\rangle}}{2^n}$
- If <**z**,**c**> = 1, the terms will cancel each out and we would get $0/2^n$. In contrast, if <**z**,**c**> = 0, the sum will be $\pm 2/2^n = \pm 1/2^{n-1}$.
- So we will only find those binary strings such that <**z**,**c**> = 0.

Quantum version (cont’d)

- Some concrete examples in the book! Pages 190-195.



Reader Tip. Warning: admittedly, working out all the gory details of an example can be a bit scary. We recommend that the less meticulous reader move on to the next section for now. Return to this example on a calm sunny day, prepare a good cup of your favorite tea or coffee, and go through the details: the effort will pay off. ♡

- In conclusion, for given periodic *f*, we can find the period **c** in *n* function evaluations. This in contrast to the $2^{n-1} + 1$ needed classically.

???

Reading

- This lecture: Ch 6.1-6.3
- Next lecture: Ch 6.4-6.5